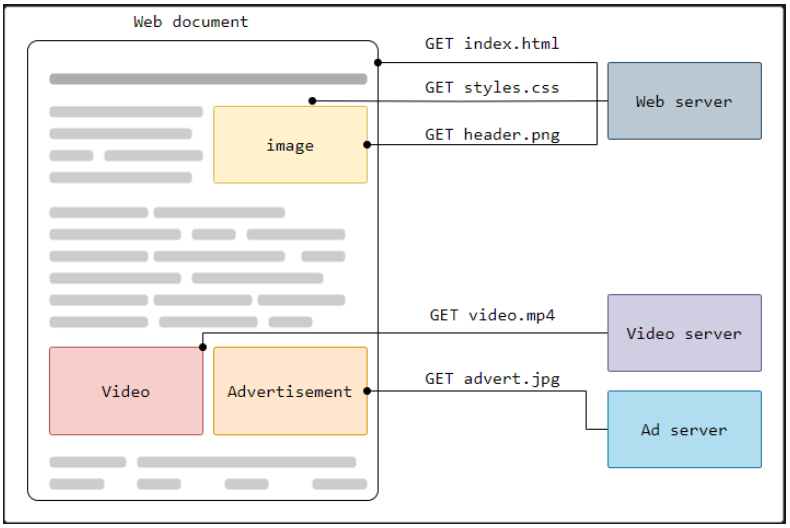
Uma visão geral do HTTP

[https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Overview]

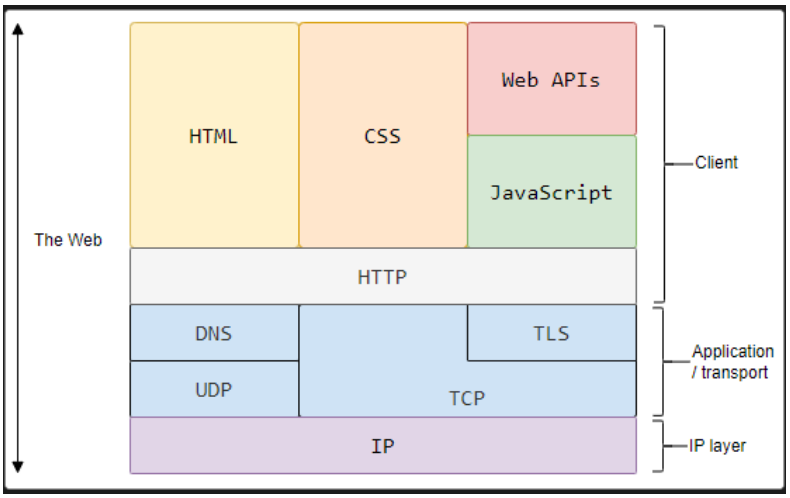
03/06/2025, 21:15

Uma visão geral do HTTP

O HTTP é um protocolo para buscar recursos como documentos HTML. Ele é a base de qualquer troca de dados na Web e é um protocolo cliente-servidor, o que significa que as requisições são iniciadas pelo destinatário, geralmente o navegador Web. Um documento completo é tipicamente construído a partir de recursos como conteúdo de texto, instruções de layout, imagens, vídeos, scripts e mais.



Clientes e servidores se comunicam trocando mensagens individuais (em oposição a um fluxo de dados). As mensagens enviadas pelo cliente são chamadas de requisições e as mensagens enviadas pelo servidor como resposta são chamadas de respostas.

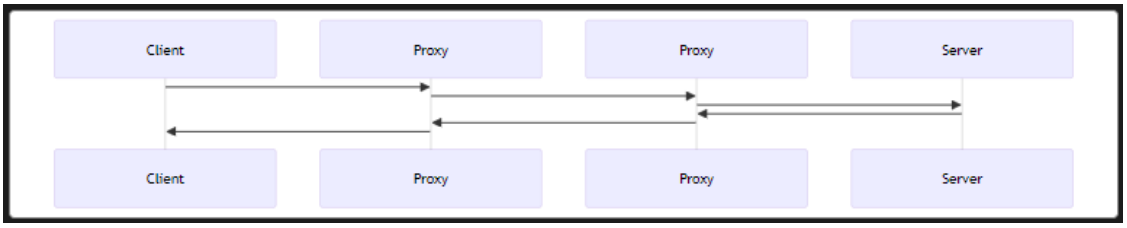


Projetado no início da década de 1990, o HTTP é um protocolo extensível que evoluiu ao longo do tempo. Ele é um protocolo de camada de aplicação que é enviado sobre TCP, ou sobre uma conexão TCP criptografada com TLS, embora qualquer protocolo de transporte confiável possa ser usado teoricamente. Devido à sua extensibilidade, ele é usado não apenas para buscar documentos de hipertexto, mas também imagens e vídeos, ou para postar conteúdo em servidores, como com resultados de formulários HTML. O HTTP também pode ser usado para buscar partes de documentos para atualizar páginas Web sob demanda.

Componentes de sistemas baseados em HTTP

O HTTP é um protocolo cliente-servidor: as requisições são enviadas por uma entidade, o agente do usuário (ou um proxy em seu nome). Na maioria das vezes, o agente do usuário é um navegador Web, mas pode ser qualquer coisa, por exemplo, um robô que rastreia a Web para preencher e manter o índice de um mecanismo de busca.

Cada requisição individual é enviada a um servidor, que a processa e fornece uma resposta chamada de resposta. Entre o cliente e o servidor há inúmeras entidades, coletivamente chamadas de proxies, que executam diferentes operações e atuam como gateways ou caches, por exemplo.



Na realidade, há mais computadores entre um navegador e o servidor que trata a requisição: há roteadores, modems e outros. Graças ao design em camadas da Web, esses elementos estão ocultos nas camadas de rede e transporte. O HTTP está no topo, na camada de aplicação. Embora sejam importantes para diagnosticar problemas de rede, as camadas subjacentes são, em sua maioria, irrelevantes para a descrição do HTTP.

Cliente: o agente do usuário

O agente do usuário é qualquer ferramenta que atua em nome do usuário. Esse papel é desempenhado principalmente pelo navegador Web, mas também pode ser desempenhado por programas usados por engenheiros e desenvolvedores Web para depurar suas aplicações.

O navegador é sempre a entidade que inicia a requisição. Nunca é o servidor (embora alguns mecanismos tenham sido adicionados ao longo dos anos para simular mensagens iniciadas pelo servidor).

Para exibir uma página Web, o navegador envia uma requisição original para buscar o documento HTML que representa a página. Em seguida, ele analisa esse arquivo, fazendo requisições adicionais correspondentes a scripts de execução, informações de layout (CSS) para exibição e sub-recursos contidos na página (geralmente imagens e vídeos). O navegador Web então combina esses recursos para apresentar o documento completo, a página Web. Scripts executados pelo navegador podem buscar mais recursos em fases posteriores, e o navegador atualiza a página Web de acordo.

Uma página Web é um documento de hipertexto. Isso significa que algumas partes do conteúdo exibido são links, que podem ser ativados (geralmente por um clique do mouse) para buscar uma nova página Web, permitindo ao usuário direcionar seu agente do usuário e navegar pela Web. O navegador traduz essas direções em requisições HTTP e interpreta as respostas HTTP para apresentar ao usuário uma resposta clara.

O servidor Web

Do lado oposto do canal de comunicação está o servidor, que fornece o documento conforme solicitado pelo cliente. Um servidor aparece apenas como uma única máquina virtualmente; mas pode, na verdade, ser um conjunto de servidores compartilhando a carga (balanceamento de carga), ou outro software (como caches, um servidor de banco de dados ou servidores de e-commerce), gerando total ou parcialmente o documento sob demanda.

Um servidor não é necessariamente uma única máquina, mas várias instâncias de software de servidor podem ser hospedadas na mesma máquina. Com HTTP/1.1 e o cabeçalho Host, elas podem até compartilhar o mesmo endereço IP.

Proxies

Entre o navegador Web e o servidor, numerosos computadores e máquinas retransmitem as mensagens HTTP. Devido à estrutura em camadas da pilha Web, a maioria deles opera nas camadas de transporte, rede ou física, tornando-se transparente na camada HTTP e potencialmente tendo um impacto significativo no desempenho.

Aqueles que operam nas camadas de aplicação são geralmente chamados de proxies. Eles podem ser transparentes, encaminhando as requisições que recebem sem alterá-las de forma alguma, ou não transparentes, caso em que eles alteram a requisição de alguma forma antes de repassá-la ao servidor. Os proxies podem desempenhar várias funções:

- **cache** (o cache pode ser público ou privado, como o cache do navegador)
- **filtragem** (como uma varredura antivírus ou controle parental)
- **balanceamento de carga** (para permitir que vários servidores atendam a requisições diferentes)
- **autenticação** (para controlar o acesso a diferentes recursos)
- **registro de logs** (permitindo o armazenamento de informações históricas)

Aspectos básicos do HTTP

HTTP é simples

O HTTP é geralmente projetado para ser legível por humanos, mesmo com a complexidade adicional introduzida no HTTP/2 ao encapsular mensagens HTTP em quadros.
As mensagens HTTP podem ser lidas e compreendidas por humanos, proporcionando testes mais fáceis para desenvolvedores e menor complexidade para iniciantes.

HTTP é extensível

Introduzidos no HTTP/1.0, os cabeçalhos HTTP tornam este protocolo fácil de estender e experimentar.
Novas funcionalidades podem até ser introduzidas por um acordo entre cliente e servidor sobre a semântica de um novo cabeçalho.

HTTP é sem estado, mas não sem sessão

O HTTP é sem estado: não há ligação entre duas requisições executadas sucessivamente na mesma conexão.
Isso tem imediatamente a perspectiva de ser problemático para usuários tentando interagir com certas páginas de forma coerente, por exemplo, usando cestas de compras de e-commerce.

Mas, enquanto o núcleo do HTTP é de fato sem estado, os **cookies HTTP** permitem o uso de sessões com estado. Usando a extensibilidade dos cabeçalhos, os cookies HTTP são adicionados ao fluxo de trabalho, permitindo a criação de sessões em cada requisição HTTP para compartilhar o mesmo contexto, ou o mesmo estado.

HTTP e conexões

Uma conexão é controlada na camada de transporte e, portanto, fundamentalmente fora do escopo do HTTP.
O HTTP não exige que o protocolo de transporte subjacente seja baseado em conexão; ele apenas exige que seja confiável, ou que não perca mensagens (no mínimo, apresentando um erro nesses casos).

Entre os dois protocolos de transporte mais comuns na Internet:

- **TCP é confiável**
- **UDP não é**

Portanto, o HTTP depende do padrão TCP, que é baseado em conexão.

Antes que um cliente e um servidor possam trocar um par de requisição/resposta HTTP, eles devem estabelecer uma conexão TCP, um processo que exige várias trocas (round-trips).

O comportamento padrão do HTTP/1.0 é abrir uma conexão TCP separada para cada par de requisição/resposta HTTP.
Isso é menos eficiente do que compartilhar uma única conexão TCP quando múltiplas requisições são enviadas em sucessão próxima.

Para mitigar essa falha, o HTTP/1.1 introduziu o **pipelining** (que provou ser difícil de implementar) e conexões persistentes:
a conexão TCP subjacente pode ser parcialmente controlada usando o cabeçalho Connection.

O HTTP/2 foi um passo adiante ao **multiplexar mensagens sobre uma única conexão**, ajudando a manter a conexão ativa (quente) e mais eficiente.

Experimentos estão em andamento para projetar um protocolo de transporte melhor e mais adequado ao HTTP.
Por exemplo, o Google está experimentando com o **QUIC**, que se baseia no UDP para fornecer um protocolo de transporte mais confiável e eficiente.

O que pode ser controlado pelo HTTP

Essa natureza extensível do HTTP permitiu, ao longo do tempo, maior controle e funcionalidade da Web.
Métodos de cache e autenticação foram funções tratadas logo no início da história do HTTP.
A capacidade de relaxar a restrição de origem, por outro lado, foi adicionada apenas na década de 2010.

A seguir, uma lista de funcionalidades comuns controláveis com HTTP:

- **Cache:**
  Como os documentos são armazenados em cache pode ser controlado pelo HTTP.
  O servidor pode instruir proxies e clientes sobre o que armazenar e por quanto tempo.
  O cliente pode instruir proxies intermediários de cache a ignorar o documento armazenado.

- **Relaxamento da restrição de origem:**
  Para evitar espionagem e outras invasões de privacidade, os navegadores Web aplicam uma separação estrita entre sites.
  Somente páginas da **mesma origem** podem acessar todas as informações de uma página Web.
  Embora tal restrição seja um fardo para o servidor, cabeçalhos HTTP podem **relaxar** essa separação estrita no lado do servidor,
  permitindo que um documento se torne um mosaico de informações provenientes de diferentes domínios; pode até haver razões relacionadas à segurança para fazer isso.

- **Autenticação:**
  Algumas páginas podem ser protegidas de modo que apenas usuários específicos possam acessá-las.
  A autenticação básica pode ser fornecida pelo HTTP, seja utilizando os cabeçalhos como WWW-Authenticate e similares,
  ou configurando uma sessão específica usando cookies HTTP.

- **Proxy e tunelamento:**
  Servidores ou clientes estão frequentemente localizados em intranets e escondem seu endereço IP verdadeiro de outros computadores.
  Requisições HTTP então passam por **proxies** para cruzar essa barreira de rede.
  Nem todos os proxies são proxies HTTP.
  O protocolo **SOCKS**, por exemplo, opera em um nível inferior.
  Outros protocolos, como FTP, podem ser tratados por esses proxies.

- **Sessões:**
  O uso de cookies HTTP permite vincular requisições com o estado do servidor.
  Isso cria sessões, apesar de o HTTP básico ser um protocolo sem estado.
  Isso é útil não apenas para cestas de compras em e-commerces, mas também para qualquer site que permita **configuração personalizada** da saída pelo usuário.

Fluxo HTTP

Quando um cliente deseja se comunicar com um servidor, seja o servidor final ou um proxy intermediário, ele executa os seguintes passos:

1. **Abrir uma conexão TCP:**
   A conexão TCP é usada para enviar uma requisição, ou várias, e receber uma resposta.
   O cliente pode abrir uma nova conexão, reutilizar uma conexão existente ou abrir várias conexões TCP com os servidores.

2. **Enviar uma mensagem HTTP:**
   As mensagens HTTP (antes do HTTP/2) são legíveis por humanos.
   Com o HTTP/2, essas mensagens são encapsuladas em quadros (*frames*), tornando-as impossíveis de serem lidas diretamente,
   mas o princípio permanece o mesmo. Por exemplo:

```http
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

3. Ler a resposta enviada pelo servidor, como:

```http
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html

<!doctype html>… (aqui vêm os 29769 bytes da página Web solicitada)
```

4. Fechar ou reutilizar a conexão para requisições posteriores.

Se o **pipelining HTTP** estiver ativado, várias requisições podem ser enviadas sem esperar pela recepção completa da primeira resposta.
O pipelining HTTP mostrou-se difícil de implementar em redes existentes, onde softwares antigos coexistem com versões modernas.

O pipelining HTTP foi substituído no HTTP/2 por uma multiplexação mais robusta de requisições dentro de um quadro.

Mensagens HTTP

As mensagens HTTP, conforme definidas no HTTP/1.1 e anteriores, são legíveis por humanos.
No HTTP/2, essas mensagens são incorporadas em uma estrutura binária, chamada de **quadro** (*frame*), permitindo otimizações como:

- compressão de cabeçalhos
- multiplexação

Mesmo que apenas parte da mensagem HTTP original seja enviada nesta versão do HTTP, a **semântica de cada mensagem permanece inalterada**
e o cliente reconstitui (virtualmente) a requisição original do HTTP/1.1.
Portanto, é útil compreender as mensagens HTTP/2 no formato do HTTP/1.1.

Existem dois tipos de mensagens HTTP:

1. **Requisições**
2. **Respostas**

Cada um com seu próprio formato.

Requisições

Exemplo de requisição HTTP:

As requisições consistem nos seguintes elementos:

- Um **método HTTP**, geralmente um verbo como GET, POST, ou um substantivo como OPTIONS ou HEAD, que define a operação que o cliente deseja executar.
  Tipicamente, um cliente deseja buscar um recurso (usando GET) ou postar o valor de um formulário HTML (usando POST),
  embora mais operações possam ser necessárias em outros casos.
- O **caminho** do recurso a ser buscado; a URL do recurso, **sem** elementos que são óbvios a partir do contexto, por exemplo, **sem o protocolo** (http://), **sem o domínio** (aqui, developer.mozilla.org), ou a **porta TCP** (aqui, 80).
- A **versão** do protocolo HTTP.
- **Cabeçalhos opcionais** que transmitem informações adicionais aos servidores.
- Um **corpo**, para alguns métodos como POST, semelhante aos das respostas, que contêm o recurso enviado.

Respostas

Exemplo de resposta:

As respostas consistem nos seguintes elementos:

- A **versão do protocolo HTTP** que elas seguem.
- Um **código de status**, indicando se a requisição foi bem-sucedida ou não, e por quê.
- Uma **mensagem de status**, uma descrição curta e não autoritativa do código de status.
- **Cabeçalhos HTTP**, como os das requisições.
- Opcionalmente, um **corpo** contendo o recurso obtido.

APIs baseadas em HTTP

Como o HTTP é um protocolo de rede muito comum, ele serve como base para muitas outras aplicações baseadas na Web.
A API mais usada baseada em HTTP é a **Fetch API**, amplamente usada para executar chamadas AJAX que foram originalmente realizadas usando a interface XMLHttpRequest.

Eventos enviados pelo servidor

Uma API baseada em HTTP que não envolve a Fetch API é a API de **eventos enviados pelo servidor** (*server-sent events*).
Neste caso, o navegador usa a interface EventSource para criar uma conexão entre o cliente e o servidor, e então o servidor pode enviar eventos para o cliente.

Estes eventos são entregues ao cliente como **mensagens HTTP**, mas são analisados, convertidos em objetos Event, e entregues automaticamente
ao manipulador de eventos do navegador (onmessage ou um manipulador como onopen ou onerror, ou um registrado com addEventListener).

Conclusão

O HTTP é um protocolo fácil de usar e extremamente extensível.
Sua natureza cliente-servidor, combinada com a habilidade de adicionar novos campos de cabeçalho facilmente, permitiu que ele evoluísse,
se adaptando às demandas crescentes da Web.

O encapsulamento de mensagens HTTP em quadros no HTTP/2 e a introdução do HTTP/3 trouxeram **maior desempenho** e **complexidade técnica** para os navegadores.
Contudo, a estrutura básica das mensagens HTTP permaneceu a mesma desde o HTTP/1.0.

O fluxo de sessão continua simples, e pode ser analisado com ferramentas básicas como sniffers de rede HTTP, depuradores de navegador ou utilitários de linha de comando como o curl.